VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Řízení anténního rotátoru

Antenna Rotator Control

Tomáš Eisner

Bakalářská práce

Vedoucí práce: Ing. David Seidl, Ph.D.

Ostrava, 2021

Abstrakt

Práce je zaměřená na implementaci programu pro mikrokontroler ESP32 určenému pro ovládání anténního rotátoru. Mikrokontroler ESP32 komunikuje s rotátorem přes rozhraní UART, pomocí kterého posílá požadavky založené na zjednodušené implementaci knihovny Hamlib. Součástí práce je také webový server založený na TCP socketech, který běží přímo na ESP32 a umožňuje ovládání rotátoru v uživatelsky přívětivějším rozhraní. Další vlastností je možnost přímého propojení mikrokontroleru ESP32 s programem GPredict, který umožňuje řízení anténního rotátoru za účelem sledování satelitů v reálném čase. Obě tyto služby běží na mikrokontroleru paralelně.

Klíčová slova

Mikrokontroler ESP32, ESP-IDF, anténní rotátor, webserver, GPredict, TCP sockety, SPIFFS, Hamlib, UART

Abstract

This thesis is focused on implementation of a software for microcontroller ESP32 to control antenna rotator. Microcontroller ESP32 communicates with the rotator via UART interface that transfers requests and commands based on simplified Hamlib communication protocol. The microcontroller is also running a webserver based on TCP sockets which provides a user-friendly web interface. Another important feature is providing an interface for GPredict client which provides the ability to track satellites and real time rotator control. Both of these features run simultaneously on the microcontroller.

Keywords

Microcontroller ESP32, ESP-IDF, antenna rotator, webserver, GPredict, TCP socket, SPIFFS, Hamlib, UART

Poděkování

Rád bych poděkoval Ing. Davidu Seidlovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Obsah

Seznam použitých zkratek a symbolů

Framework – struktura sloužící jako nástroj pro programování a vývoji software
IoT – Internet of Things, internet věcí.
IDE – integrované vývojové prostředí, používá se k vývoji software
ESP-IDF – oficiální vývojový framework pro vývoj aplikací pro ESP32
API – rozhraní pro programování aplikací, sbírka knihoven a funkcí
JSON – JavaScript Object Notation
SPIFFS – Seriál Peripheral Interface Flash File Systém

1 Úvod

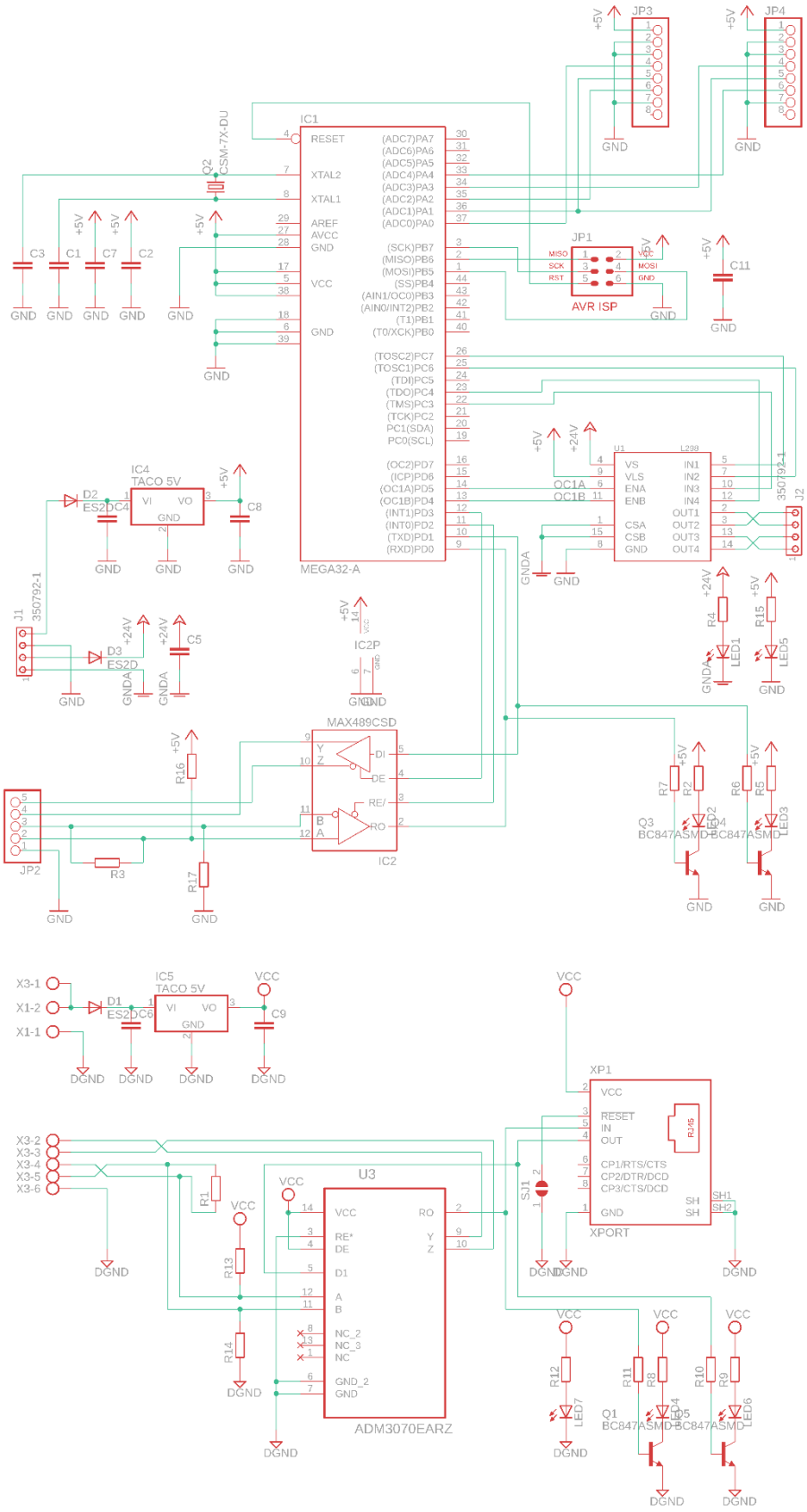
Antény, převážně pak všesměrové antény, jsou nedílnou součástí každodenní komunikace a běžného života. Setkáváme se s nimi prakticky všude, od rádií, mobilních telefonů až po bezdrátové routery. V mnoha případech však všesměrové antény nejsou dostatečně výkonné a je třeba jejich výkon směřovat jedním konkrétním směrem, proto jsou často používány takzvané směrové antény. Směrové antény vyzařují nebo přijímají signál pouze v určitém směru, což zvyšuje sílu signálu a redukuje interferenci s okolními zdroji. Pro účely, kde je dosah a síla signálu nezbytná se proto používají směrové antény s vysokým ziskem. Ty jsou typické pro odvětví jako jsou například armádní, letecký nebo kosmický průmysl. Obzvláště v těchto odvětvích je třeba maximální přesnosti a minimální chybovosti nasměrování. Korekce a neustálá kalibrace je ovšem časově náročná a je tedy vhodné tyto úkony automatizovat. Mikrokontrolery a mikroprocesory, které se pro ovládání chodu a zaměřování takovýchto zařízení využívají, jsou tedy nezbytné.

Obsluha a práce s takovými zařízeními je však technicky náročná a nepraktická. Programátoři a technický personál jsou tak velmi často jediní, kdo je schopen takto sestavená zařízení využívat a spravovat. Nedílnou součástí automatizace je tak přizpůsobení uživatelského rozhraní a funkcionality i pro běžné koncové uživatele bez pokročilé technické, resp. programátorské znalosti.

2 Seznámení s funkcionalitou rotátoru

Rotátor směrové antény, se kterým v této práci pracuji, je poháněn dvěma stejnosměrnými komutátorovými motory pro vertikální a horizontální otáčení. Pro správné a přesné ovládání motorů je na každé hřídeli umístěn magnetický detektor natočení, který detekuje aktuální pozici hřídele. Motory jsou pak napojeny na silovou elektroniku, která se stará o zapínání a vypínání obou motorů a řídí směr jejich otáčení. Celá funkcionalita rotátoru je řízená mikroprocesorem Atmel. Ten je zodpovědný za komunikaci s detektory natočení a zajišťuje ovládání silové elektroniky. Schéma anténního rotátoru lze pozorovat na obr. 2.1. Tato kapitola má sloužit pouze k uvedení do problematiky rotátoru a poskytnutí základních informací. Vnitřní stavbou a návrhem jsem se v rámci této práce nezabýval, jelikož se tato práce zaměřuje pouze na vstupně/výstupní komunikaci s rotátorem a na jeho využití. Bližší podrobnosti a technické detaily vzhledem k profilaci nebudou v této práci uvedeny.

Rotátor navrhl a sestavil Ing. David Seidl, Ph.D., který mi jej zapůjčil pro zhotovení této práce a následné praktické testování.



Obrázek č. 2.1 – schéma rotátoru

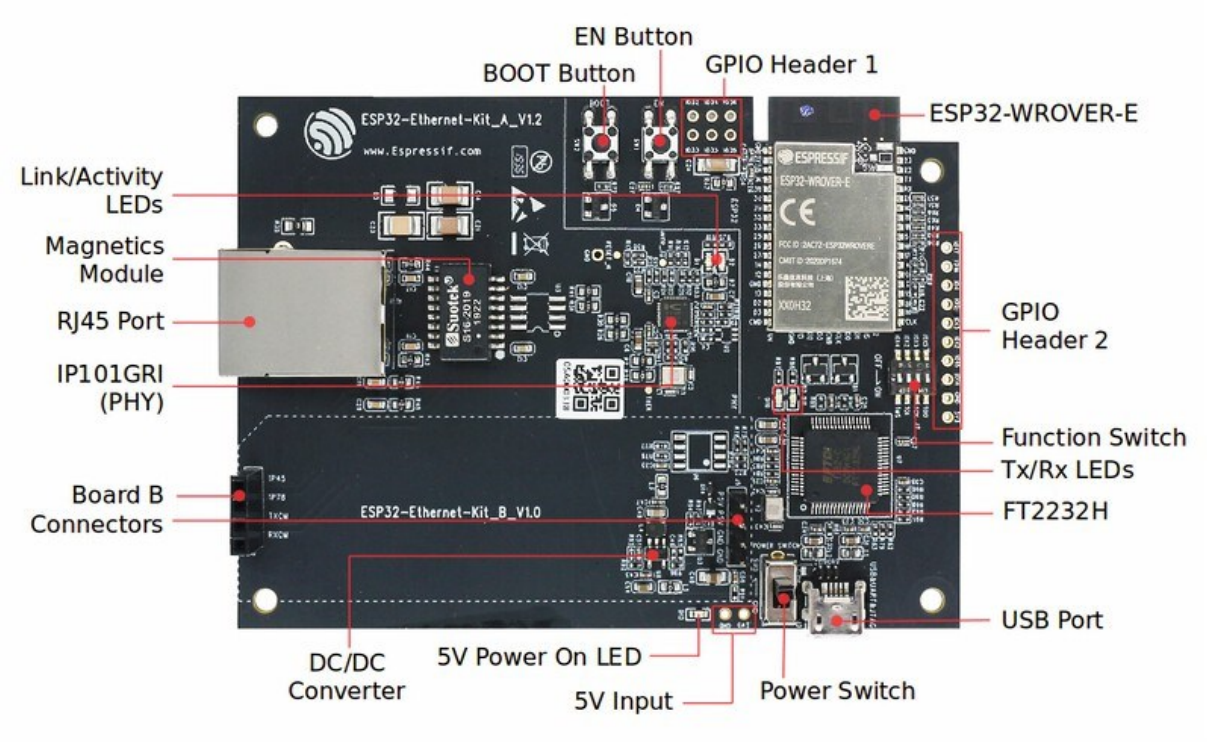
Obrázek č. 2.2 – fotografie rotátoru

3 Programování mikrokontroleru ESP32

3.1 Mikrokontroler ESP32

ESP32 je cenově dostupný mikrokontroler vytvořený společností Espressif Systems, který díky automatizaci a rozšiřování zájmu o IoT získává stále větší popularitu. Existuje mnoho variant modulů a vývojových desek s odlišnými specifikacemi, které tento mikrokontroler obsahují. Pro tuto práci jsem zvolil vývojovou desku *ESP32-Ethernet-Kit*.

Tato varianta obsahuje ESP32-WROVER-E modul se zabudovaným Bluetooth a Wi-Fi, IP101GRI Ethernetovým transceiverem s jedním 10/100 portem. Modul se připojuje pomocí USB rozhraní, konkrétně mikro USB konektorem. Přítomnost obou variant, Wi-Fi i klasického RJ45 konektoru pro ethernet byla jednou z hlavních důvodů proč byla tato varianta použita.



Obrázek 3.1.1: ESP32-Ethernet-Kit schéma [5]

Na desce se nachází dostatečné množství univerzálních vstupně-výstupních pinů (GPIO), které umožňují samotné propojení vývojové desky s anténním rotátorem.

Další nezbytnou součástí tohoto modulu je zabudovaný debugger přímo na desce. Debugger pomohl vyladit problematické části aplikace a pomohl vyřešit problémy, které by byly jinak jen stěží pozorovatelné.

3.2 Programovací prostředí

Pro programování existuje několik možností. Mezi nejpopulárnější patří bezesporu Arduino IDE, které po doinstalování rozšíření pro podporu ESP32 a ESP-IDF framework umožňuje programovat pomocí tohoto prostředí a jeho programovacího jazyka. Dalšími možnostmi jsou například MicroPython, což je reimplementace programovacího jazyka Python 3 pro embedded zařízení, JavaScript, LUA nebo oficiální ESP-IDF framework.

MicroPython je velmi zajímavá volba, která stejně jako Python pro běžné počítače v mnohém ulehčuje vývoj. Nevýhodou je však nižší výkon, jelikož ESP32 musí kromě napsaného programu ještě obsahovat interpret Pythonu. Osobně se také cítím jistější při programování v C než v Pythonu, proto jsem tuto možnost nezvolil.

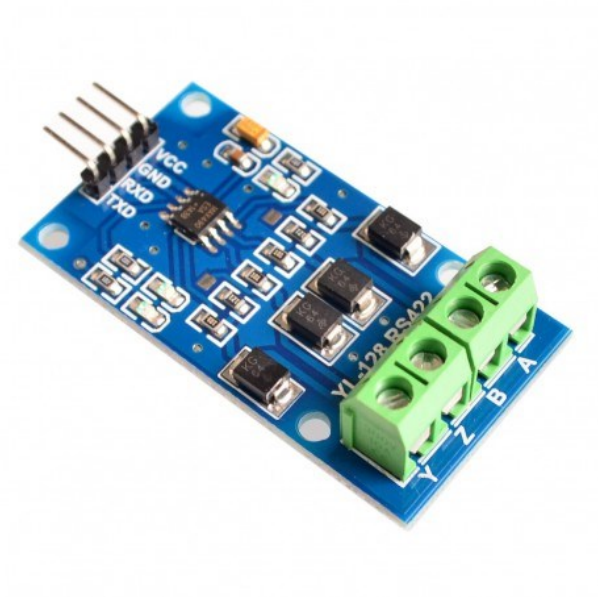
Arduino IDE zaobaluje funkcionalitu ESP-IDF a přizpůsobuje vlastnímu prostředí. Toto IDE by mělo být užíváno přesně ze stejných důvodů a stejnými lidmi, jako užívají arduino vývojové desky – kvůli jednoduchosti a uživatelsky přívětivému rozhraní, které vyžaduje relativně malé množství znalostí. Dostupnost návodů a podpory je v tomto případě značná. Mezi další výhody patří například portabilita mezi různými mikrokontrolery. Nevýhodou oproti ESP-IDF je nižší výkon nebo horší rozšiřitelnost kódu, tudíž nepraktičnost pro větší projekty. Z programátorského hlediska Arduino IDE svádí k přístupu vše programovat v jedné hlavní smyčce, i přesto, že by program měl být dělen na menší celky a z nich být složen. Pod obalem se sice program takto rozděluje, právě díky rozšířením pro ESP32 knihovny, ale nad programem nemůžu mít přímou kontrolu. Tento fakt také znamená, že Arduino IDE je stavěno převážně na práci s jedním jádrem.

Espressif IDF (ESP-IDF) je oficiální vývojový framework pro rodinu ESP-32 mikrokontrolerů a je to také prostředí, které jsem použil pro tuto práci. Jelikož se jedná o oficiální framework, dokumentace je velmi podrobná a obsahuje všechny potřebné informace. Díky tomu, že je tento framework oficiální jsem měl jistotu, že veškeré funkce, které by mohly být potřebné a které bych mohl využít budou dostupné. Přímý přístup k ovládanému hardware bez nutnosti další nástavby tak umožňuje preciznější rozdělení projektu a přesné nastavení přes *menuconfig*, což je jednoduchý nástroj pro konfiguraci vytvářeného projektu. Velkou výhodou je také bezproblémová podpora freeRTOS a dostupnost všech aktualizací a nových funkcí ihned po jejich vydání. Plná podpora využití obou jader ESP32 je pak samozřejmostí.

K programování jsem použil textový editor Visual Studio Code s rozšířením pro podporu ESP-IDF, tudíž samotné psaní kódu bylo podstatně příjemnější. Toto rozšíření navíc umožňuje i nahrávat, sestavovat, debutovat či konfigurovat vytvořený projekt což výrazně ulehčuje a zrychluje celý proces. Nevýhoda tohoto rozšíření je však občasná disfunkce našeptávání referencí hlavičkových souborů a tudíž se stává, že některé funkce se ukazují jako nedostupné, i přesto že po sestavení projektu je vše bez chyby.

4 Zapojení

Anténní rotátor má jako výstupní rozhraní pro komunikaci RS-422. Pro zapojení a zprovoznění komunikace s mikrokontrolerem ESP32 přes rozhraní UART je třeba využít obousměrný plně duplexní konvertor RS-422 na UART v napěťových úrovních TTL, které jsou s mikrokontrolerem kompatibilní.



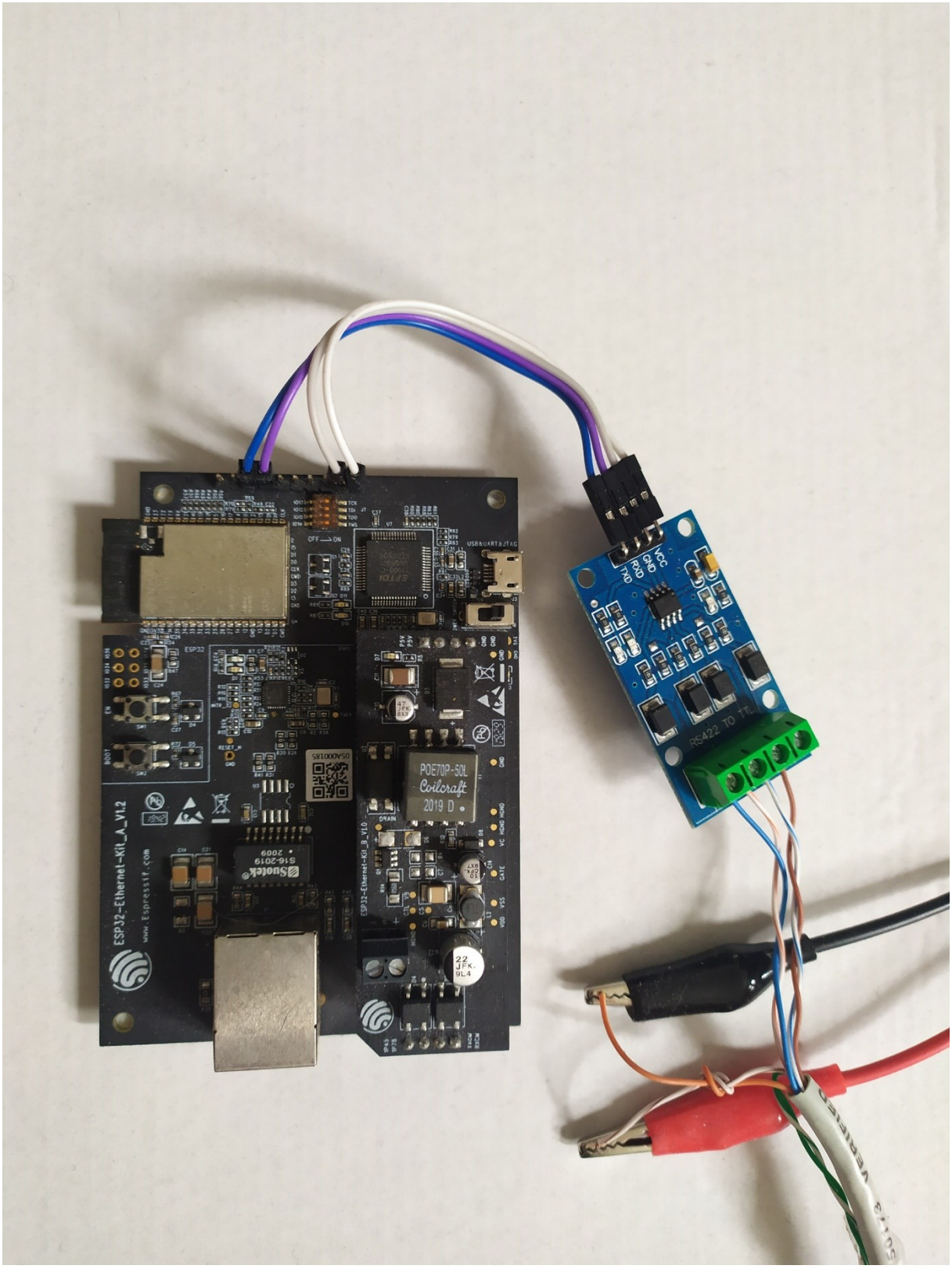
Obrázek č. 4.1 – RS-422-TTL konvertor [6]

Na mikrokontroleru ESP32 byly využity univerzální vstupně/výstupní piny GPIO4 pro TXD a GPIO2 pro RXD. Pro připojení bylo nezbytné využít také piny pro napájení a zem, aby komunikace přes UART fungovala správně. Podle technické dokumentace jsem dále ke konvertoru připojil samotný rotátor. Pro napájení rozhraní RS-422 a anténního rotátoru jsem využil napájecí zdroj s výstupním napětím 12 V a maximálním možným výstupním proudem 2000 mA.

Mikrokontroler ESP32 je s konvertorem propojen dle následující tabulky.

Tabulka č. 4.1 – Tabulka zapojení konvertoru

Pin mikrokontroleru ESP32	Pin konvertoru
GPIO4	TXD
GPIO2	RXD
GND	GND
3V3	VCC



Obrázek č. 4.2 – Reálné zapojení

5 Ovládání rotátoru mikrokontrolerem ESP32

Požadavky na ovládání rotátoru jsou poměrně jasné – je třeba zajistit otáčení jak horizontálně, tak vertikálně. Ovládání rotátoru, které je naprogramováno v interním procesoru Atmel, funguje přes sériovou linku na rozhraní RS-422. Směrem do mikrokontroleru ESP32 proudí data o současné poloze a opačným směrem se posílají příkazy, které rotátor ovládají. Ovládání však funguje pouze způsobem, že po obdržení příkazu se rotátor točí daným směrem, dokud není příkazem stop zastaven.

5.1 Komunikace přes rozhraní UART

Samotná komunikace mikrokontroleru ESP32 s rotátorem probíhá přes rozhraní UART. V překladu univerzální asynchronní přijímač/vysílač je fyzický obvod, který slouží k přenosu dat přes sériovou linku. Jak už název napovídá, UART je asynchronní, nemá tedy hodinový signál a je dáno strukturou dat kde zpráva začíná a kde končí. Pro přenos dat se používají dva datové signály, RXD, který slouží k přijímaní dat a TXD, který data odesílá. Struktura přenášených dat je dána konfigurací daného sériového portu kde lze nastavit například přenosovou rychlost, paritu, počet přenesených bitů nebo stop bit.
V prostředí ESP-IDF je konfigurace rozhraní UART poměrně jednoduchá. Oficiální knihovna *„driver/uart.h“* obsahovala vše co jsem potřeboval. Nejprve je třeba vytvořit požadované nastavení.

```
const uart_config_t uart_config = {
    .baud_rate = 9600,
    .data_bits = UART_DATA_8_BITS,
    .parity = UART_PARITY_DISABLE,
    .stop_bits = UART_STOP_BITS_1,
    .flow_ctrl = UART_HW_FLOWCTRL_DISABLE,
    .source_clk = UART_SCLK_APB,
};
```

Ukázka kódu č. 5.1.1 – UART základní konfigurace

Následně je nezbytné nainstalovat ovladač pro rozhraní UART na požadovaný port. Port rozhraní UART si lze zvolit libovolně, v mém případě *UART_NUM_0 – UART_NUM_2. RX_BUF_SIZE* je konstanta pro množství bitů, které lze přes rozhraní UART přijmout.

```
uart_driver_install(UART_NUM_1, RX_BUF_SIZE * 2, 0, 0, NULL, 0);
```

Ukázka kódu č. 5.1.2 – Konfigurace ovladače

Nyní už stačí pouze přiřadit vytvořené nastavení k vybranému portu rozhraní UART a následně tomuto portu přiřadit zvolené piny na mikrokontroleru ESP32 pro RXD a TXD.

```
uart_param_config(UART_NUM_1, &uart_config);
uart_set_pin(UART_NUM_1,TXD_PIN,RXD_PIN,UART_PIN_NO_CHANGE);
```

Ukázka kódu č. 5.1.3 – Konfigurace pinů pro UART

Tímto je rozhraní připraveno k posílání i přijímání dat a lze s ním dále pracovat pomocí funkcí z oficiální knihovny ESP-IDF. [11]

5.2 Knihovna pro otáčení rotátoru

Aby bylo možné rotátor otáčet precizně na požadované souřadnice, bylo třeba pro ovládání rotátoru napsat malou knihovnu funkcí. Knihovna obsahuje pouze základní funkce nezbytné ke správnému fungování všech komponent práce a pracuje s programem na interním procesoru Atmel, který má implementovanou základní funkcionalitu pro otáčení rotátoru. Implementované instrukce, které procesor Atmel očekává na svém UART rozhraní, lze vidět v tabulce č. 5.2.1. Jednořádkové instrukce jsou ukončeny znakem nového řádku *„\n”.*

Tabulka č. 5.2.1 – instrukce interního procesoru

Instrukce	**Akce**
„r“	*Otáčení vpravo*
„l“	*Otáčení vlevo*
„u“	*Otáčení nahoru*
„d“	*Otáčení dolů*
„ “	*Zastavení*

První dvě funkce knihovny, znázorněné v ukázce kódu č. 5.2.1, slouží k nastavení horizontální a vertikální pozice s požadovanou pozicí předanou jako parametr ve stupních. Nezbytným parametrem je také číslo portu rozhraní UART, který jsme pomocí oficiální knihovny ESP-IDF *driver/uart.h* vybrali v předchozí podkapitole.

$$void\ turn\_deg\_v\ (uart\_port\_t\ uart\_num, float\ v\_dest\_deg);$$

$$void\ turn\_deg\_h\ (uart\_port\_t\ uart\_num, float\ h\_dest\_deg);$$

Ukázka kódu č. 5.2.1 – Funkce otáčení

Napřed dojde k načtení současné pozice rotátoru, kterou posílá rotátor každých 100ms přes UART. Takto načtená data následně rozdělím a získám požadované hodnoty, tedy horizontální a vertikální pozici. Zmíněnou funkcionalitu můžeme vidět v ukázce kódu č. 5.2.2.

$$uart\_read\_bytes(uart\_num, data, BUF\_SIZE, 100\ /\ portTICK\_RATE\_MS);$$

$$sscanf((int*)data, "\%*[^H]H:\%4d(\%f)\ V:\%4d(\%f)", \&temp1, \&h\_deg, \&temp2, \&v\_deg);$$

Ukázka kódu č. 5.2.2 – Načtení a zpracování dat

Rotátor podporuje oboustranné otáčení. Aby se rotátor neotáčel kolem své osy o více než 180° a otáčel se správným směrem, byly tyto funkce optimalizovány pro snížení opotřebení motorů a zrychlení otáčení.

```
#Pro vertikální otáčení
if ( h_deg < h_dest_deg ){
    d = "d\n";
}
else d = "u\n";

#Pro horizontální otáčení
if (v_dest_deg > v_deg && (v_dest_deg - v_deg) <= 180) {
    d = "d\n";
}
else d = "u\n";
```

Ukázka kódu č. 5.2.3 – Optimalizace otáčení

Otáčet lze s přesností na 2 stupně, viz ukázka kódu č. 5.2.4. Toto omezení by bylo možné lehce vylepšit, ale z důvodu občasných nepřesností v detekci pozice, případně v otáčení motoru, jsem nastavil 2° jako ideální hodnotu, které zamezí možným problémům a neočekávanému chování.

```
if ( abs( deg - dest_deg ) > 2 ) {
        uart_write_bytes(uart_num, (const char*) d, 2);
}
else {
        uart_write_bytes(uart_num, (const char*) " \n", 2);
        free(data);
        break;
}
```

Ukázka kódu č. 5.2.4 – Otáčení rotátoru

Posledními dvěma funkcemi z mnou vytvořené knihovny jsou funkce pro zjištění aktuální polohy rotátoru.

```
float get_radius_h(uart_port_t uart_num);
float get_radius_v(uart_port_t uart_num);
```

Ukázka kódu č. 5.2.5 – Funkce zjištění natočení

6 GPredict software

GPredict je opensource aplikace, která v reálném čase predikuje polohu velkého množství satelitů a jejich orbit. Aplikace přehledně zobrazuje jejich podrobné souřadnice a jiná data v různých formátech [1]. Jednou z mnoha výhod je také schopnost předpovídat přesný čas a souřadnice průletu satelitů a poskytnout o každém z nich detailní informace. Je nutno podotknout, že GPredict v současné době umí pracovat pouze s orbitami země, meziplanetární orbity jsou plánovány pro budoucí verze programu.
V současné době pracuji s verzí GPredict 2.2.1.

6.1 Základní princip fungování programu

Funkčnost programu a jeho přesné a rychlé sledování satelitů v reálném čase zajišťují algoritmy NORAD SGP4/SDP4. [2] Jedná se o zjednodušené matematické modely užívané ke kalkulaci orbitálních stavových vektorů satelitů a vesmírného odpadu a prachu v závislosti na Inerciálním souřadnicovém systému země. [3] Tyto algoritmy následně vypočítávají jeden z Kepplerových zákonů [12], konkrétně rovnici o pohybu na orbitě. Program následně provádí několik korekcí pro kompenzaci přírodních jevů, zakřivení země a gravitace. [4]

6.2 Knihovna Hamlib a komunikační protokol pro ovládání anténních rotátorů

Hamlib [7] je knihovna pro ovládání rádií a rotátorů převážně určená pro amatérské uživatele, která je z většiny psána v jazycích C a C++, ale podporuje několik dalších skriptovacích jazyků jako Perl, Python nebo Lua. Jedná se o open-source projekt, který je podporován a rozšiřován desítkami lidí, kteří se zasluhují o vytvoření softwarové vrstvy pro ovládání široké škály běžně dostupných zařízení. Nejběžnější užívání této knihovny je však pro ovládání rádií nebo rotátorů přes TCP/IP protokol. Hamlib využívá dva démony pro TCP/IP, *rotctld a rigctld*, které slouží jako nádstavba a rozhraní pro C knihovnu, se kterou komunikují na základě přijatých textových příkazů od uživatele.

Pro tuto práci je nejdůležitější z těchto démonů *rotctld, implementován v Linuxu,* který umožňuje ovládání anténních rotátorů. Tento démon komunikuje s klientem přes TCP socket pomocí jednořádkových příkazů. Požadavky si GPredict vyměňuje jakožto klient s TCP serverem implementovaným u druhé strany, tedy na mém mikrokontroleru ESP32. Jelikož se jedná o vyrobený anténní rotátor, nikoliv o běžný kus, který je volně dostupný ke koupi, nebylo možné využít linuxovou implementaci knihovny Hamlib, ale bylo nutné knihovnu implementovat na mikrokontroleru ESP32. GPredict má ve své dokumentaci zdokumentovány pouze základní funkce nezbytné pro ovládání rotátoru. V mé práci byly potřeba pouze dvě, ty nejzákladnější, ukázány v ukázce kódu č. 6.2.1.

```
P, set_pos 'Azimuth' 'Elevation'
p, get_pos

Ex.:  P 170 45 -> požadavek o nastavení pozice rotátoru
```

Ukázka kódu č. 6.2.1 – Hamlib komunikační protokol

7 Webové rozhraní pro ESP32

7.1 SPI FLASH FILE SYSTEM – SPIFFS

Ukládání webových HTML, CSS a JavaScript souborů na mikrokontroleru není triviální záležitost. Mnoho veřejně dostupných řešení webových serverů pro mikrokontrolery ukládá celé soubory jako řetězce a následně načítá jako webovou stránku. Takovýto přístup je velice nepraktický a obtížně se s ním pracuje. Mnohem lepší řešení je tedy využít SPIFFS.

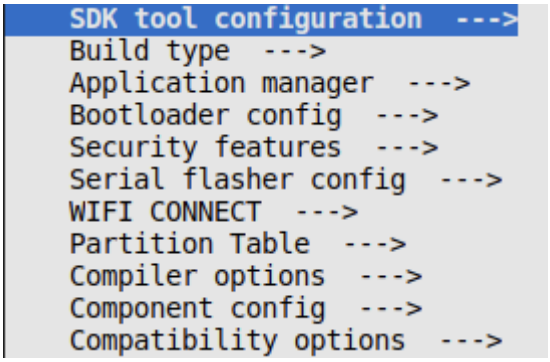
Jedná se o nenáročný souborový systém pro mikrokontrolery, který pomocí SPI rozhraní (sériové periferní rozhraní) pro komunikaci mezi procesorem a jinými integrovanými obvody komunikuje v tomto případě s pamětí flash. Se souborovým systémem lze pracovat jako s běžnými soubory pomocí utilit jazyka C. SPIFFS umožňuje používat flash paměť na mikrokontroleru, jako by to byl souborový systém, umožňuje tak pracovat se soubory v klasické podobě, a tím zlepšit vývoj webové aplikace.

SPIFFS je třeba nakonfigurovat a je nezbytné alokovat dostatečné množství paměti. To je nutné nakonfigurovat v tabulce diskových oddílů (partition table). Tabulka diskových oddílů není běžně součástí aplikace pro mikrokontroler ESP32 a je tedy nutné tento soubor vytvořit a přidat. Obsah souboru lze vidět v ukázce kódu č. 7.1.1.

```
nvs,     data, nvs,    ,      0x6000,
phy_init, data, phy,    ,      0x1000,
factory,  app,  factory, ,      1M,
spiffs,  data,  spiffs , ,      1M,
```

Ukázka kódu č. 7.1.1 – Tabulka diskových oddílů

Následně je nutné správně tyto nové diskové oddíly nastavit, aby byly při boot sekvenci správně načteny. Toho lze docílit přes *menuconfig* (obrázek č. 7.1.1), což je jedna ze základních utilit mikrokontrolerů ESP32. Slouží ke konfiguraci základních parametrů mikrokontroleru ESP32, nastavení ethernet připojení, WiFi, bluetooth, velikost paměti flash, a mnoho dalších.



Obrázek č. 7.1.1 – menuconfig

Nakonec je nutné v souboru CMakeLists.txt vytvořit partition image pomocí následujícího příkazu.

```
spiffs_create_partition_image(spiffs ../html FLASH_IN_PROJECT)
```

Ukázka kódu č. 7.1.2 – Vytvoření diskového oddílu

7.2 Webový server na mikrokontroleru ESP32

Samotný webový server běží v samostatném vlákně a čeká na příchozí spojení od klienta. Toto přiřazení funkce k vláknu se provádí v „main“ funkci pomocí základních metod knihovny ESP-IDF. Mimo jiné se zde zadává funkce, která na vláknu bude spuštěna, pracovní název a také paměť alokovaná pro vlákno.

```
xTaskCreate(&OnConnected, "btsis", 1024 * 5, NULL, 5, NULL);
```

Ukázka kódu č. 7.2.1 – Přiřazení funkce vláknu při spuštění

Jedná se o velmi jednoduchý server, který je nakonfigurován dle výchozích parametrů, které knihovna *esp_http_server.h* poskytuje. Tato struktura webového serveru byla z velké části inspirována oficiálním ukázkovým projektem [9] webového serveru pro mikrokontroler ESP32 od Espressif Systems a veřejně dostupným projektem [8] na platformě GitHub. Tato dvě řešení obsahují funkčnost, kterou jsem od webového serveru očekával, a tudíž nebylo nutné základní webový server vymýšlet nanovo.

```
httpd_handle_t server = NULL;
httpd_config_t config = HTTPD_DEFAULT_CONFIG();
config.uri_match_fn = httpd_uri_match_wildcard;



ESP_LOGI(TAG, "starting server");
if (httpd_start(&server, &config) != ESP_OK)
{
   ESP_LOGE(TAG, "COULD NOT START SERVER");
}
```

Ukázka kódu č. 7.2.2 – Základní konfigurace webového serveru

Následně pomocí SPIFFS a struktury *httpd_uri_t* byly vytvořeny koncové body webového serveru, které uživatel může navštívit a zasílat na ně požadavky. Jejich vytvoření je popsán v ukázce kódu č. 7.2.3.

```
httpd_uri_t dir_end_point_config = {

   .uri = "/dir",

   .method = HTTP_POST,

   .handler = on_dir_set};

httpd_register_uri_handler(server, &dir_end_point_config);
```

Ukázka kódu č. 7.2.3 – Konfigurace koncových bodů na webovém serveru

Webový server umožňuje oboustrannou komunikaci s mikrokontrolerem ESP-32, resp. s rotátorem. Komunikace na straně webového serveru probíhá pomocí asynchronních funkcí v programovacím jazyce JavaScript a je znázorněna níže.

```
                                                    ##Pro zasílání dat do mikrokontroleru

async function sendDir(element_id){

  fetch("/dir", {method:"POST", body: JSON.stringify({"direction": element_id})});
  console.log(element_id);
  console.log(JSON.stringify({"direction": element_id}));
}
                                                    ##Pro přijímání dat z mikrokontroleru

    async function getCoordinates(){
      const result = await fetch("/coordinates");
      const h = await result.json();

      $('#elTxt').val(h.el);
      $('#azTxt').val(h.az);
    }


    setInterval(function () { if(!set){getCoordinates();} }, 2000);

    async function sendGpredictEnabled(value){

      fetch("/gpredict_enable", {method:"POST", body: JSON.stringify({"gpredict":
      value})});
    }


    async function sendCoordinates(){

      let el = $('#elTxt').val();
      let az = $('#azTxt').val();
      fetch("/dirCustom", {method:"POST", body: JSON.stringify({"el": el, "az": az})});

      $('#send_coordinates').css("visibility", "hidden");
    }
```

Ukázka kódu č. 7.2.4 – Komunikace na straně webového serveru

Na straně webového serveru dochází k periodickému načítání pozice natočení anténního rotátoru, které probíhá každé 2 vteřiny.

Na straně mikrokontroleru byly použity metody z knihovny *cJSON.h,* které jsem využil pro zpracování dat v následujících funkcích, díky kterým lze data mezi webserverem a mikrokontrolerem vyměňovat.

```
static esp_err_t on_dir_set(httpd_req_t *req);
static esp_err_t on_hello(httpd_req_t *req);
```

Ukázka kódu č. 7.2.5 – Komunikace na straně mikrokontroleru

Přijímání dat je realizováno pomocí přijetí http požadavku na koncovém bodu */dir* a následnému převedení z formátu JSON na standardní řetězec jazyku C.

```
httpd_req_recv(req, buf, req->content_len);
cJSON *rcv = cJSON_Parse(buf);
cJSON *direction = cJSON_GetObjectItem(rcv, "direction");
char *str = cJSON_GetStringValue(direction);
```

Ukázka kódu č. 7.2.6 – Základní konfigurace webového serveru

S takto získanou hodnotou lze dále pracovat a zasílat rotátoru odpovídající instrukce.
Předávání informací směrem k webserveru je realizováno pomocí zaslání http odpovědi, která je vyvolána požadavkem na koncový bod */data.* Jako odpověď jsou zaslány aktuální souřadnice natočení anténního rotátoru.

```
httpd_resp_send(req, message, strlen(message));
```

Ukázka kódu č. 7.2.7 – Odpověď mikrokontroleru

Pro otáčení anténního rotátoru na konkrétní pozici byl vytvořen další koncový bod */dirCustom*.
Na tento koncový bod jsou zasílány požadované souřadnice, které jsou následně zpracovány obdobným způsobem jako u manuálního otáčení. Funkce však navíc otáčí anténní rotátor na požadované pozice pomocí metod z mnou vytvořené knihovny.

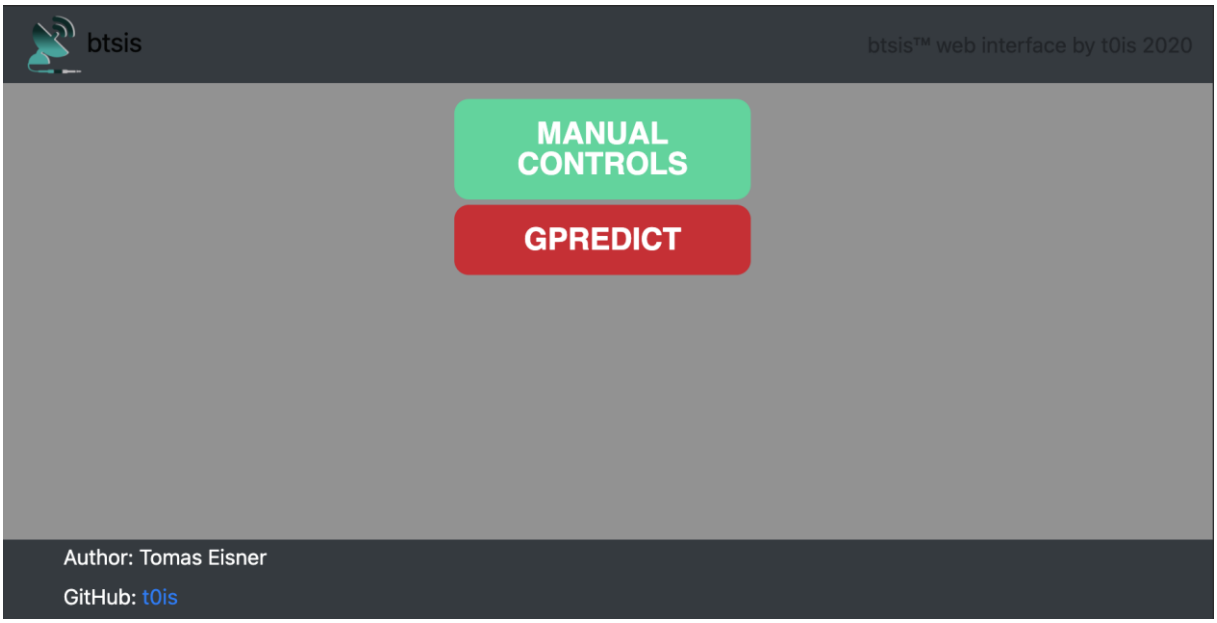
```
cJSON *rcv = cJSON_Parse(buf);
cJSON *el_json = cJSON_GetObjectItem(rcv, "el");
cJSON *az_json = cJSON_GetObjectItem(rcv, "az");
char *el = cJSON_GetStringValue(el_json);
char *az = cJSON_GetStringValue(az_json);

turn_deg_h(UART_NUM_1, atof(az));
turn_deg_v(UART_NUM_1, atof(el));
```

Ukázka kódu č. 7.2.7 – Odpověď mikrokontroleru

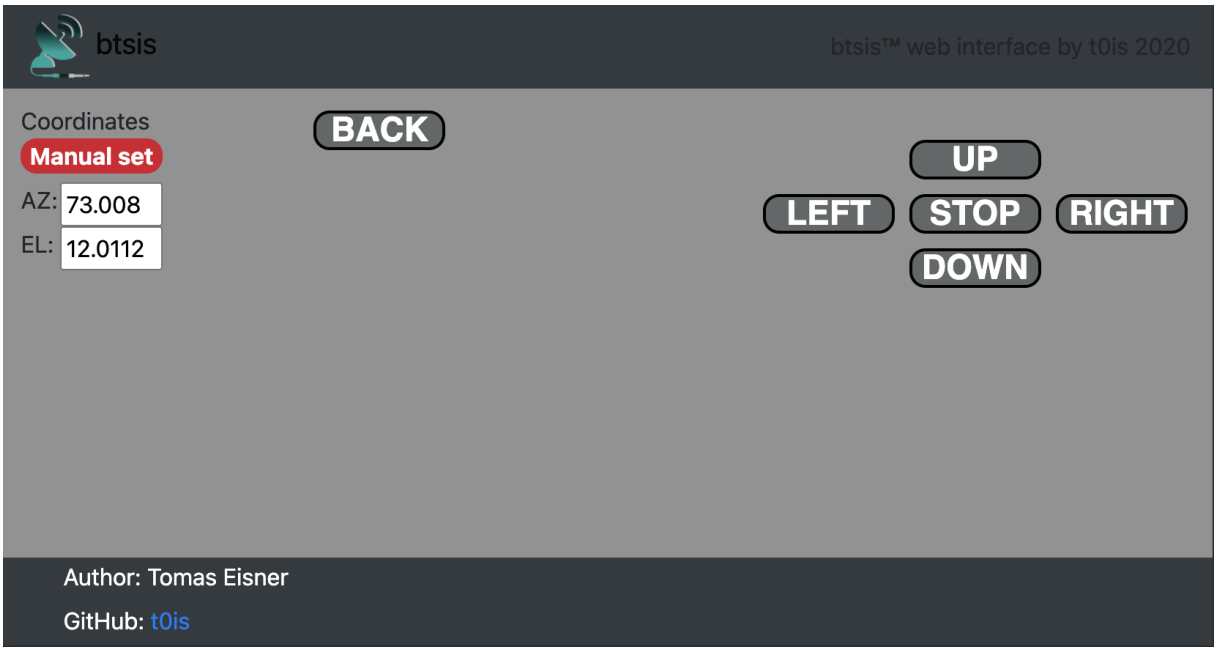
7.3 Návrh webové stránky a její funkce

Webová stránka obsahuje základní funkce pro ovládání anténního rotátoru. Mým hlavním cílem při jejím vytváření byla jednoduchost a jasná funkcionalita. Grafický design není mou silnou stránkou, tudíž je grafický návrh stránky spíše symbolický a zjednodušený.
Na webové stránce, kterou můžeme vidět na následujícím obrázku, se nachází dvě hlavní tlačítka, které aktivují rozšiřující funkce na základě toho, které z nich je aktivováno.

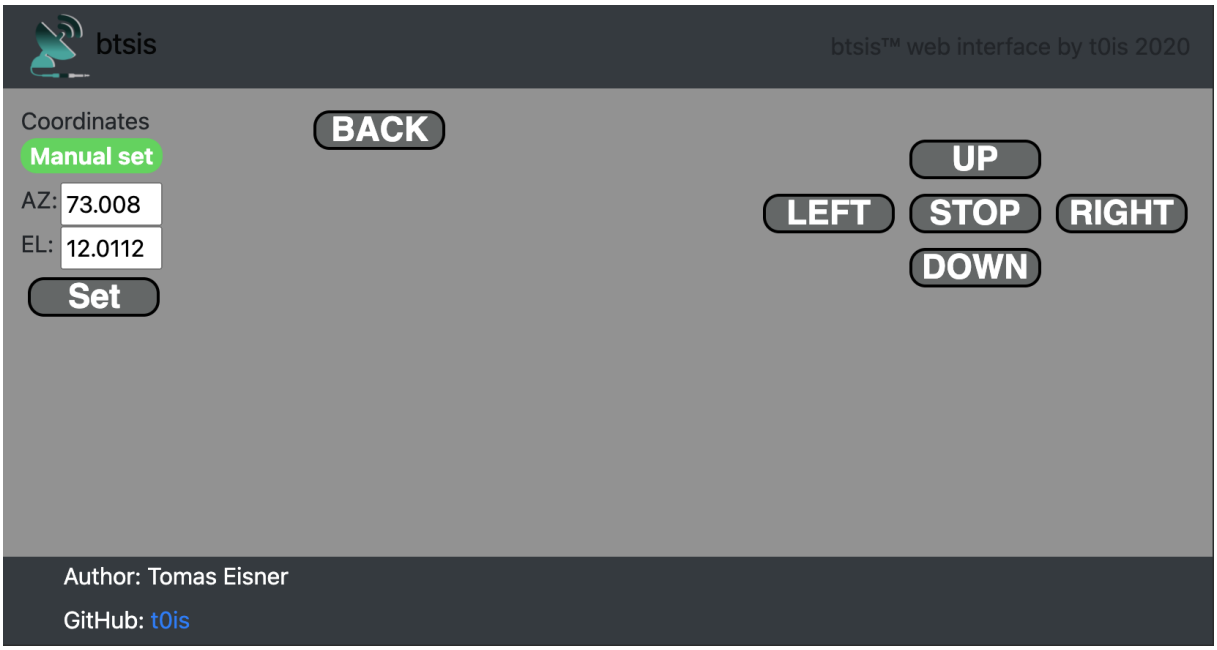


Obrázek č. 7.3.1 – Webová stránka úvod

První tlačítko *MANUAL CONTROLS* spouští manuální režim ovládání. Po zmáčknutí směrového tlačítka se anténní rotátor začne otáčet příslušným směrem. Nutno podotknout, že pro zahájení rotace stačí tlačítko zmáčknout, není třeba jej držet po celou dobu. Pro zastavení rotace slouží tlačítko STOP.



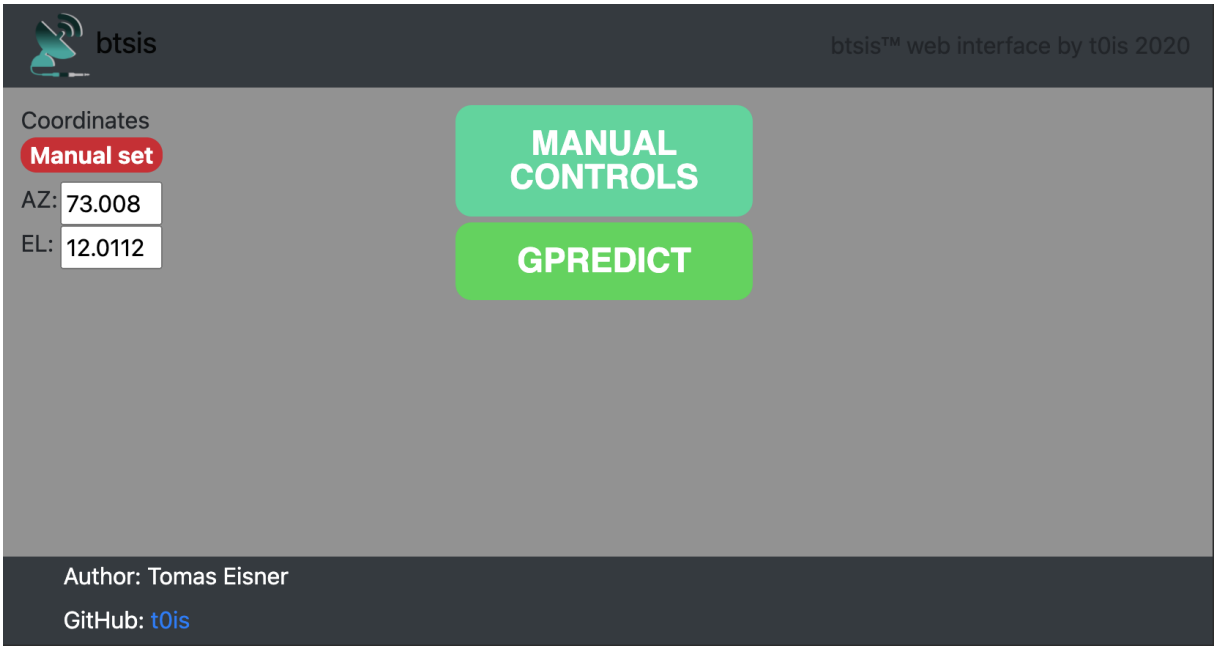
Obrázek č. 7.3.2 – Webová stránka manuál



Obrázek č. 7.3.3 – Webová stránka manuál 2

Webové rozhraní také podporuje natočení na zadané souřadnice. Po zmáčknutí tlačítka *Manual set* se pozastaví automatické čtení pozice a po zadání konkrétních koordinátů na levé straně webové stránky, 0 až 360 pro azimut, 0 až 90 pro elevaci, a zmáčknutí tlačítka *Set* dojde k natočení anténního rotátoru na požadovanou pozici

Druhé tlačítko *GPREDICT* aktivuje automatický režim ovládání natočení rotátoru, který je spravován programem GPredict. Toto tlačítko umožní připojení klienta GPredict k severu na mikrokontroleru ESP32. Na levé straně jsou opět dvě pole se současnými souřadnicemi natočení anténního rotátoru. V tomto módu ovšem nejsou souřadnice editovatelné, veškeré ovládání je řízeno výhradně přes GPredict software. Aktivování tlačítka *Manual set* pak deaktivuje automatické řízení programem GPredict.



Obrázek č. 7.3.4 – Webová stránka GPredict

8 Propojení mikrokontroleru ESP32 s programem GPredict

Pro práci s GPredict softwarem je nutné zajistit několik nezbytných věcí. Jako první věc je důležité zprovoznit sockety. Pomocí socketů vytvoříme server, na který se GPredict jakožto klient následně připojí. Po navázání spojení je nezbytné správně zpracovat přijatá data a současně zasílat programu GPredict aktuální data o poloze natočení anténního rotátoru. To obnáší využití více vláken, aby k těmto úkonům mohlo docházet současně.

8.1 Socket server na mikrokontroleru ESP32

Socket server je nezbytnou součástí této práce, jelikož realizuje spojení s GPredict softwarem, jakožto klienta, a umožňuje tak plně automatický režim zaměřování anténního rotátoru.

Server využívající sockety je součástí ukázkových projektů [9] v ESP-IDF. Jedná se o velmi jednoduchý server, což ale mým požadavkům naprosto vyhovuje, jelikož pro tento úkon není nic komplexnějšího třeba. Tento server vytvoří socket na kterém poslouchá a čeká na spojení s klientem. Po navázání spojení server alokuje pro navázané spojení port a na něm poslouchá a čeká na příjem dat. Jestliže server data přijme, spustí funkci pro následnou práci s GPredict softwarem.

Server na mikrokontroleru ESP32 následně načte data přijatá od klienta, od GPredict software.

```
len = recv(sock, rx_buffer, sizeof(rx_buffer) - 1, 0);
rx_buffer[len] = 0;
double az, el;
char* buf_pointer = rx_buffer;
buf_pointer = str_replace(buf_pointer, ",", ".");
sscanf(buf_pointer, "P %lf %lf", &az, &el);
```

Ukázka kódu č. 8.1.1 – Zpracování dat z GPredict

Z ukázky kódu lze poznat, že je nutné formát přijatých dat trochu upravit. GPredict obsahuje jednoduchou implementaci knihovny Hamlib a využívá pro zaslaná data čísla s desetinnou čárkou, zatímco v programovacím jazyce C se využívá desetinná tečka. Zároveň program GPredict pro přijímání dat ze serveru očekává desetinnou tečku. Tato vlastnost je poměrně zvláštní a neočekávaná, ale pomocí jednoduché funkce na úpravu přijatého řetězce není problém jej upravit do požadovaného formátu, který je vhodný pro další použití. Programovací jazyk C neobsahuje funkci, která by tuto úpravu umožňovala, žádná funkce typu replace(), které známe z vyšších programovacích jazyků, není v jazyku C k dispozici. Bylo tedy nutné vytvořit vlastní funkci.

```
char* str_replace(char* orig, char* rep, char* new);
```

Ukázka kódu č. 8.1.2 – Funkce replace

Tato funkce přijímá jako parametr původní řetězec, původní znak, který má být nalezen a nový znak, který má všechny původní znaky nahradit. Funkce byla inspirována řešením na veřejném fóru stackoverflow [10] a doplněna dle manuálových stránek funkcí *memcpy, memset* a *strcpy*.

8.2 Paralelní provoz webového serveru a socket serveru pro GPredict

Pro zajištění současného provozu obou služeb současně, tedy webového serveru pro ovládání rotátoru a serveru, na který se připojuje GPredict jakožto klient, bylo nutné správně konfigurovat paralelní funkcionalitu, která byla pro tuto práci žádoucí. V první řadě bylo nutné v *main* funci na mikrokontroleru přiřadit funkce různým vláknům.

```
xTaskCreate(&OnConnected, "btsis", 1024 * 5, NULL, 5, NULL);
xTaskCreate(&Gpredict_server, "btsis_gpredict", 1024 * 5, NULL, 5, NULL);
```

Ukázka kódu č. 8.2.1 – Přidělení funkcí vláknům

Tato metoda z knihovny FreeRTOS [13] zajistí spuštění separátních vláken pro parametrem zvolené funkce. Provoz GPredict serveru vyžadoval následně ještě další vlákno, jelikož server by měl umět současně provádět požadovanou rotaci a posílat programu GPredict informace o současné poloze. V tomto případě jsem využil POSIX vlákna, se kterými mám větší zkušenosti.

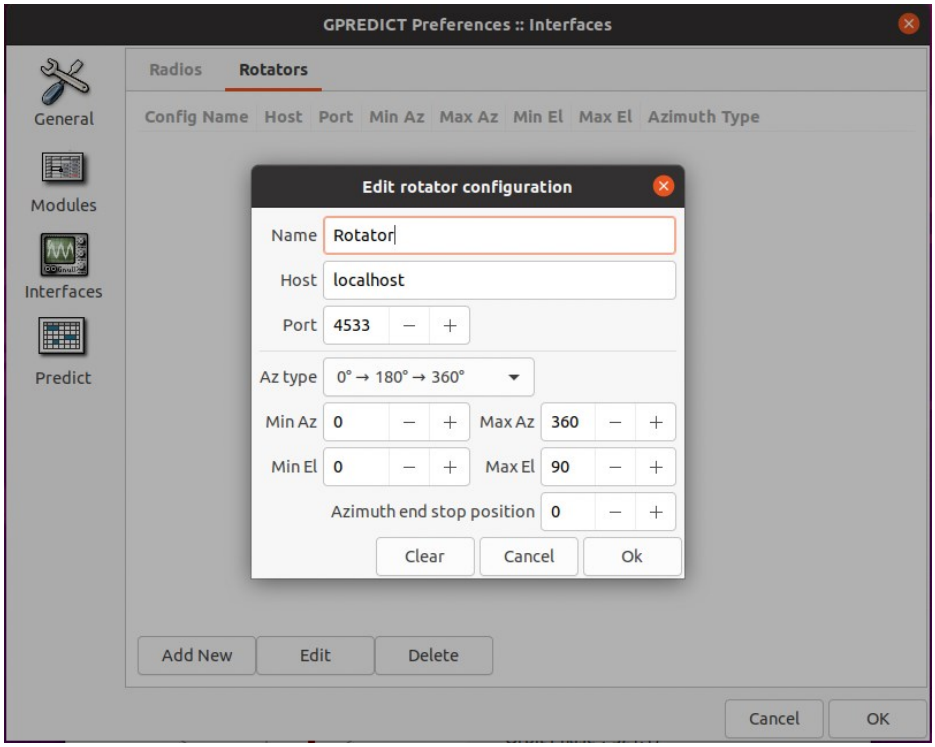
```
void* thread_turn_h(void* args){
  struct pt_args* p = args;
  turn_deg_h(UART_NUM_1, p->value_h);
  turn_deg_v(UART_NUM_1, p->value_v);

  ESP_LOGE(TAG, "thread %f %f", p->value_h, p->value_v);
  return NULL;
}
```

Ukázka kódu č. 8.2.2 – GPredict paralelismus

8.3 Konfigurace anténního rotátoru a používání GPredict software

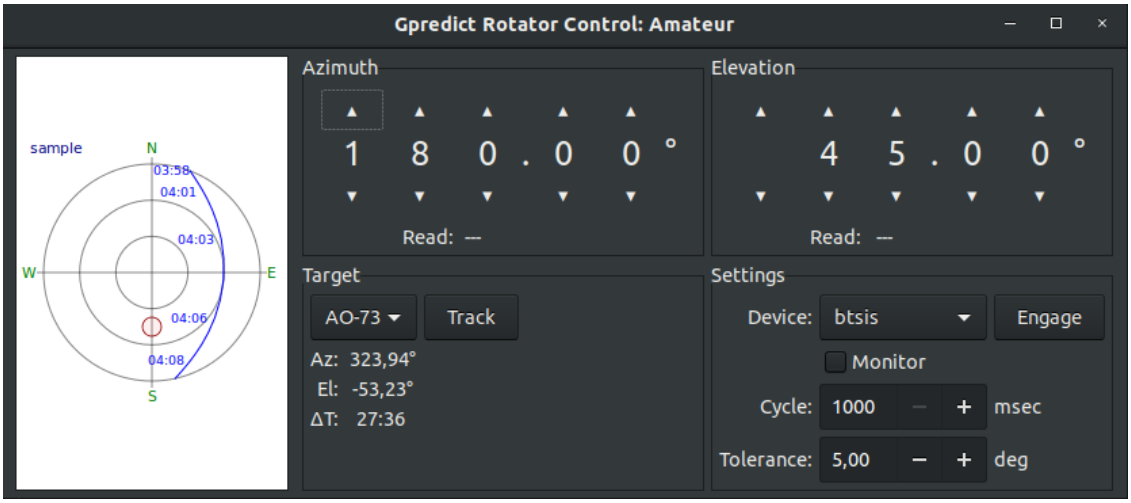
Po nainstalování programu GPredict je nutné nakonfigurovat anténní rotátor jako nové rozhraní, což se provádí v preferencích programu.



Obrázek č. 8.3.1 – GPredict konfigurace

Jméno anténního rotátoru je libovolné, host je IP adresa, kterou rotátor obdržel na lokální síti a port je komunikační port, který je nakonfigurován přímo v mém programu a může být určen libovolně. Pro tuto práci jsem zachoval výchozí nastavení a použil port 4533. Pro ostatní parametry jsem použil výchozí hodnoty. Takto nakonfigurovaný anténní rotátor lze nyní použít, GPredict se nyní jako klient může připojit k serveru, který běží na mikrokontroleru ESP32.

Následně je už jednoduché vybrat konkrétní satelit a pomocí jednoho tlačítka zapnout automatický režim, kdy pomocí předem zmíněné konfigurace GPredict ovládá natočení anténního rotátoru.



Obrázek č. 8.3.2 – Ovládací panel GPredict

9 Závěr

V rámci bakalářské práce bylo zadáno navrhnout a realizovat program pro mikroprocesor ESP32, který umožní ovládání anténního rotátoru přes webové rozhraní a zároveň umožní připojení klienta programu GPredict pro automatické ovládání natočení na pozici vybraného satelitu. Dle zadání a pomocí RS-422-TTL konvertoru jsem provedl zapojení anténního rotátoru k mikrokontroleru ESP32. Pomocí vlastní vytvořené knihovny pro ovládání anténního rotátoru jsem byl schopen s přesností na dva stupně ovládat natočení anténního rotátoru. Tato knihovna mi pomohla jednoduše ovládat veškeré potřebné funkce pro tuto práci. Knihovna by do budoucna mohla být vylepšena a doplněna o lepší práci s vlákny, kde by každou rotaci řídilo vlastní vlákno v rámci knihovny a tato funkcionalita tak nemusela být řešena v rámci programu. Kalibrací motorů by bylo možné dosáhnout lepší přesnosti natočení.

Webové rozhraní, které webový server umožňuje je velmi jednoduché, ale splňuje veškeré potřeby této bakalářské práce. Umožňuje velmi intuitivní ovládání bez nutnosti jakékoliv další znalosti, což je žádoucí. Bohužel intuitivní ovládání příliš nedoplňuje grafický design webového rozhraní, který by určitě bylo vhodné v budoucnu přebudovat, aby bylo více uživatelsky přívětivé. Funkčnost webového rozhraní je však velmi dobrá a bezproblémová, žádné chyby se nevyskytují a ovládání natočení anténního rotátoru funguje přesně tak, jak bylo požadováno. Technické řešení webového serveru pomocí SPIFFS se ukázalo být velmi spolehlivé a vhodné pro projekt této velikosti a věřím, že i s nárůstem komplexity programu by toto řešení obstálo velmi dobře.

Požadovaného propojení mikrokontroleru ESP32 s programem GPredict bylo rovněž dosaženo, což umožňuje automatické ovládání otáčení. Na mikrokontroleru ESP32 jsem vytvořil server, který očekává připojení klienta GPredict a pomocí práce s několika vlákny bylo dosaženo simultánní obsluhy veškerých služeb tohoto programu. Funkce, kterou jsem v této práci implementovat nemohl byla možnost automatického zaměřování na satelity i bez aktivace GPredict klienta, ale program GPredict bohužel žádnou API nebo knihovnu na tuto funkcionalitu v současnosti nemá.

V rámci dalšího možného vývoje by bylo vhodné implementovat funkce ovládání pře webový server a program GPredict pro různé typy rotátorů a umožnit lepší portabilitu kódu pro různé zařízení. Program by dále měl umožňovat připojení GPredict klienta na uživatelem zvolený port, namísto současného fixního výchozího portu. V neposlední řadě by pak bylo dobré doplnit program o možnost připojení pomocí ethernetového kabelu namísto Wi-Fi z důvodu stability a rozšíření možností použití.

Celá tato práce je publikována veřejně na platformě GitHub a dostupná k dalšímu vývoji. Věřím, že vzhledem k množství řešené problematiky může sloužit jako dobrý startovní bod pro různé další projekty na platformě mikrokontroleru ESP32.

Seznam příloh

Příloha v IS EDISON, praktická část práce, zdrojové kódy.

Literatura

[1] GPredict. *GPredict: Free, real-time satellite tracking and orbit prediction software for Linux, Mac OS X and Windows* [online]. London: Alexandru Csete, 2021 [cit. 2021-02-24]. Dostupné z: http://GPredict.oz9aec.net

[2] COMPARISON AND DESIGN OF SIMPLIFIED GENERAL PERTURBATION MODELS [online]. San Luis, 2009. Dostupné z: https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1094&context=theses. Diplomová práce. California Polytechnic State University, Obispo.

[3] https://www.projectpluto.com/sat_code.htm: C/C++ code for the SGP4/SDP4 satellite motion model, and for manipulating TLEs (Two-Line Elements) [online]. World: projectpluto, 1990 [cit. 2021-02-24]. Dostupné z: https://www.projectpluto.com/sat_code.htm

[4] GPredict user manual. GPredict: Free, real-time satellite tracking and orbit prediction software for Linux, Mac OS X and Windows [online]. World: Alexandru Csete, 2017 [cit. 2021-02-24]. Dostupné z: https://github.com/csete/GPredict/tree/master/doc/notes

[5] ESP32-Ethernet-Kit V1.2 Getting Started Guide. ESP32-Ethernet-Kit V1.2 Getting Started Guide [online]. [cit. 2021-03-02]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-ethernet-kit.html

[6] RS422 Bidirectional Full Duplex Two-Way TTL Signal Conversion Module [online]. World: hacktronics, [cit. 2021-03-03]. Dostupné z: https://hacktronics.co.in/rs232485-usb-ttl-converters/max490-rs422-bidirectional-full-duplex-two-way-ttl-signal-conversion-module

[7] Ham Radio Control Library [online]. [cit. 2021-03-22]. Dostupné z: https://hamlib.github.io

[8] 15_Internet_Server [online]. [cit. 2021-03-25]. Dostupné z: https://github.com/Mair/esp32-course/tree/master/_15_Internet_Server

[9] Example esp-idf projects [online]. [cit. 2021-03-28]. Dostupné z: https://github.com/espressif/esp-idf/tree/master/examples/

[10] What function is to replace a substring from a string in C? [closed] [online]. [cit. 2021-03-30]. Dostupné z: https://stackoverflow.com/questions/779875/what-function-is-to-replace-a-substring-from-a-string-in-c

[11] ESP32 ESP-IDF latest documentation [online]. [cit. 2021-04-08]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

[12] Kepplerovy zákony [online]. [cit. 2021-04-08]. Dostupné z: http://home.zcu.cz/~kehar/astrokoutek/slovnik/slovnik1.html

[13] FreeRTOS dokumentace [online]. [cit. 2021-04-08]. Dostupné z: https://www.freertos.org/features.html